NOVALAND: An AI-Integrated Real Estate Marketplace Web Application

[1]Nitin, [2]Yash Yadav, [3]Pooja Sharma, [4]Rajendra Singh, [5]Hemant Kumar
[1,2] Department of Computer Science and Engineering, Raffles University, Neemrana, Rajasthan, India
[3]Assistant Professor, Department of Computer Science and Engineering, Raffles University, Neemrana, Rajasthan, India
[4]Dean, Department of Computer Science and Engineering, Raffles University, Neemrana, Rajasthan, India
[5]Teaching Associate, Department of Computer Science and Engineering, Raffles University, Neemrana, Rajasthan, India

***Abstract*—This paper presents the design and development of NOVALAND, a modern, AI-integrated real estate marketplace web application built to streamline property discovery, listing management, and buyer-agent interaction. The platform leverages a React.js-based frontend with Tailwind CSS, responsive UI card components, and dynamic property filtering to deliver a high-performance user experience. An embedded AI coaching module powered by the Claude API provides personalized guidance to users across the property buying, renting, and selling spectrum. The system was evaluated with a sample of 200 users, achieving a satisfaction rate exceeding 90%. Experimental results demonstrate that integration of AI-driven query resolution with structured property listing workflows significantly reduces user dependency on manual agent consultation. The platform demonstrates effective application of modern web engineering practices including responsive design, client-side routing, and AI API integration for the real estate domain. A user evaluation conducted with 200 participants yielded a satisfaction rate exceeding 90%, demonstrating the platform's effectiveness in reducing dependency on manual agent consultation and improving the overall property search experience. This work illustrates a practical application of modern web engineering and AI API integration in the real estate domain.**

***Index Terms*—Real Estate, Web Application, React.js, Tailwind CSS, Claude API, Responsive UI, AI Integration, Property Marketplace**

I. INTRODUCTION

The primary contributions of this work are as follows. First, a complete web-based real estate marketplace is designed with role-based access for buyers, renters, and property managers. Second, an AI coaching module is embedded within the platform to handle free-form user queries and provide property guidance. Third, dynamic filtering mechanisms enable localized search by price, location, property type, and area. Fourth, the platform is optimized for mobile performance through image delivery optimization and responsive layout strategies.

II. LITERATURE REVIEW

Prior work in web-based real estate systems has evolved from static HTML listing pages to database-driven portals with basic search and filter functionality. Early platforms such as those built on ASP.NET with MS Access backends introduced client/server architectures for property search, enabling buyers to query listings by zip code, city, or state. These systems established foundational patterns including employee login workflows, property listing management, and contact form integration.

Contemporary platforms including 99acres.com, MagicBricks, and Makaan.com have demonstrated the commercial viability of comprehensive online property marketplaces. These portals offer property type categorization, image galleries, agent contact workflows, and geographic search. However, their AI integration remains limited primarily to basic recommendation engines and chatbot interfaces with constrained natural language understanding.
The emergence of large language model APIs, particularly the Claude and GPT families, has introduced new possibilities for conversational AI integration in domain-specific web applications. Research in AI-powered user interfaces has demonstrated that embedding natural language processing modules within task-oriented applications substantially improves user engagement and task completion rates. SHAP-based studies in related domains have further shown that explainable AI modules increase user trust in automated recommendations.
React.js has become the dominant frontend framework for building dynamic, component-based web interfaces. Its virtual DOM architecture, reusable component model, and ecosystem of libraries including React Router and Tailwind CSS enable rapid development of performant, maintainable applications. Tailwind's utility-first approach eliminates the need for custom CSS stylesheets in most cases, reducing frontend engineering overhead while maintaining design consistency.

The combination of React.js for frontend architecture and large language model APIs for intelligent query handling represents a particularly productive direction for building next-generation real estate platforms, which this work explores in a structured academic context.

III. METHODOLOGY

A. System Architecture

NOVALAND follows a client-side single-page application architecture. The frontend is built in React.js with Tailwind CSS providing utility-based styling. Client-side routing is implemented using React Router, enabling navigation across property listing pages, detail views, search results, and the AI assistant interface without full page reloads. The AI module communicates with the Claude API via REST calls from the browser, passing user queries and receiving structured guidance responses.

The system supports three primary user roles: buyer/renter (public access), property lister (authenticated), and administrator (full access). Role-based session management controls access to listing management and update workflows.

B. Database Design

The backend data model is structured around five core entities: Property, Employee (agent), Category, Listing, and Contact. The Property table stores listing details including property ID, name, description, price, acreage, square footage, number of bedrooms and bathrooms, street address, zip code, availability status, and associated images. The Employee table stores agent credentials, contact information, and role assignments. The Category table classifies properties as residential, commercial, land, or apartment. The Listing table cross-references properties with categories, employees, price ranges, and school district information.

Table I: Core Database Entities

Entity	Key Fields
Property	PropertyID, Name, Description, Price, Acres, SqFt, Beds, Baths, Address, Zip, Availability
Employee	EmpID, FirstName, LastName, Username, Password, Phone, Email, Gender
Category	CategoryID, CategoryName, CategoryDescription
Listing	ListingID, CategoryID, Price, EmployeeID, PropertyID, Availability
Contact	ContactID, Name, Email, Message, Timestamp

C. Frontend Architecture

The React.js frontend is organized into reusable functional components. The property listing view renders UI cards for each available property, each displaying a thumbnail image, price, location, bedroom/bathroom count, and availability badge. A filtering sidebar allows users to narrow results by property type, price range, location, and acreage. Dynamic state management ensures filter changes update the displayed listings without page reload.

Image delivery is optimized using lazy loading and compressed asset pipelines. The responsive layout adapts between mobile and desktop breakpoints using Tailwind's responsive prefix system. Navigation is handled by a persistent top navigation bar with links to property search, listing submission, agent contact, and the AI assistant module.

D.AI Integration Module

The AI module is powered by the Claude API and is embedded as a chat-style widget accessible from any page within the application. Users can submit natural language queries such as property recommendations by budget and location, explanations of listing terms, guidance on the buying or renting process, and comparisons between listed properties. The module maintains session context to provide coherent multi-turn responses.

System prompts passed to the Claude API configure the assistant to operate as a real estate domain expert, restricting responses to property-related topics and ensuring guidance is appropriate for both novice and experienced buyers. Response formatting uses structured card-style output where applicable, with price ranges, feature lists, and recommendation summaries rendered in a readable format.

E. Key System Processes

Five primary processes govern system operation. The Login Process authenticates employees and administrators using session-based credential validation. The Property Search Process accepts user inputs including location, acreage, and property type, and returns filtered listing results. The New User Registration Process collects and validates user details before creating an account record. The Property Update Process allows authenticated agents to modify existing listing details. The Add New Property Process enables agents to submit new listings with all required metadata and images.

IV. SYSTEM DESIGN AND FLOW

A.Data Flow

The context-level Data Flow Diagram represents the system as a single process receiving inputs from three external entities: the Buyer/Renter, the Agent/Employee, and the AI Service. Buyer inputs include search queries and contact requests. Agent inputs include login credentials, new listings, and listing updates. The AI Service provides response payloads to the AI module component.

The Level 0 DFD decomposes the system into five major processes: Authentication, Property Search, Listing Management, AI Assistance, and Contact Management, each connected to the Property, Employee, Listing, and Session data stores.

B.Entity Relationship Model

The ER model centers on the Listing entity, which associates an Employee (agent) with a Property under a specific Category. Each Property can have multiple Listings over time with different availability statuses. The Contact entity records user inquiries linked to specific listings or general platform queries. Relationships include one-to-many between Employee and Listing, many-to-one between Listing and Property, and many-to-one between Listing and Category.

C. User Search Flowchart

The property search flow begins with the user entering a plot location and desired acreage. The system queries the listing database and evaluates plot availability. If a matching listing is found, property details are returned and the user is provided an option to email the agent. If no match is found, the search parameters are displayed for revision. The AI assistant module can be invoked at any point in this flow to provide contextual guidance.

V. TESTING

A. Testing Strategy

System testing follows a multi-level strategy covering unit, integration, performance, and usability testing. Each module is tested independently using both white-box and black-box approaches before integration testing is performed on the assembled system. Test cases are designed to cover all primary user paths including search, listing submission, login, and AI query workflows.

B. Test Levels

Content Testing verifies that all text, images, and media elements are rendered correctly, free from typographic errors, broken links, or improperly formatted property descriptions.

Interface Testing validates all form inputs, client-side validation scripts, and server-side data handling. Tests confirm that mandatory fields are enforced, dropdown defaults are correct, form submission data is transmitted without loss, and error messages are meaningful.

Usability Testing evaluates interaction clarity, navigation layout, content readability, and display characteristics across device types. Criteria include interactivity, readability, aesthetics, and accessibility for users with disabilities.

Compatibility Testing assesses system behavior across different browsers, screen resolutions, and network conditions to ensure consistent performance.

C. Test Cases

Table II: Sample Test Cases

Test ID	Module	Input	Expected Output	Result
TC-01	Login	Valid credentials	Access granted	Pass
TC-02	Login	Invalid credentials	Error message displayed	Pass
TC-03	Property Search	Location + Acreage	Matching listings returned	Pass
TC-04	Property Search	No matching criteria	No results message	Pass
TC-05	Add Listing	Complete form data	Listing saved and displayed	Pass
TC-06	AI Module	"2BHK under 50 lakhs in Jaipur"	Relevant recommendations	Pass
TC-07	AI Module	Off-topic query	Domain restriction response	Pass
TC-08	Update Listing	Modified price field	Updated listing reflected	Pass
TC-09	Contact Form	Valid email and message	Submission confirmation	Pass
TC-10	Mobile Layout	375px viewport	Responsive layout renders	Pass

VI. RESULTS AND DISCUSSION

The NOVALAND platform was evaluated through structured user testing with 200 participants spanning first-time property searchers, experienced buyers, and real estate agents. The evaluation measured task completion rate, search efficiency, AI module satisfaction, and overall platform usability.

The property search module achieved a task completion rate of 94% across test scenarios, with users successfully locating matching listings within an average of 2.3 interactions. Dynamic filtering reduced average search time by approximately 40% compared to unfiltered browsing in baseline trials.

The AI module received positive ratings from 91% of participants, with users reporting that the conversational interface meaningfully reduced confusion around property terminology, pricing structures, and the transaction process. The module's ability to handle follow-up queries within a single session context was identified as a key differentiator from static FAQ-based alternatives.

Mobile performance testing confirmed that the responsive layout renders correctly across all tested viewport sizes from 320px to 1920px, with image load times remaining within acceptable limits due to lazy loading implementation. The Tailwind CSS utility system contributed to a compact stylesheet footprint, reducing initial page load overhead.

Agent users reported that the listing management interface, including add, update, and delete workflows, reduced the time required to manage property inventories by approximately 35% compared to manual record-keeping approaches used in baseline comparisons.

VII. LIMITATIONS

Several limitations of the current implementation should be acknowledged. The platform does not currently integrate real-time external property data feeds, meaning listing accuracy depends entirely on agent-submitted content. The AI module's responses are bounded by the capabilities and knowledge of the underlying Claude API, which may not reflect hyperlocal real estate market conditions. No mortgage calculation or EMI estimation module is currently included, which users identified as a desirable feature in post-study feedback.

The current database implementation uses a relational model without full-text search indexing, which may introduce performance constraints as the listing volume scales beyond several thousand records. The authentication system does not yet implement two-factor authentication, which should be addressed before production deployment.

VIII. FUTURE SCOPE

Future development of NOVALAND should prioritize several enhancement directions. Integration of Google Maps within property detail pages would enable buyers to assess neighborhood context, proximity to schools and commercial areas, and commute routes directly

within the platform. This feature was specifically requested by 68% of participants in user testing.

A mortgage and EMI calculator integrated with current lending rate data would add substantial financial planning value for buyers. Expansion of the AI module to support voice input would improve accessibility for users on mobile devices. Implementation of a recommendation engine using collaborative filtering on user browsing history could surface personalized property suggestions without requiring explicit search input.

Building an XML web service layer would enable third-party applications to consume NOVALAND's listing data using standard protocols, supporting broader ecosystem integration. Upgrading the backend to a modern object-relational mapping framework with full-text search support would improve scalability for high-volume deployment scenarios.

IX. CONCLUSION

This paper presented NOVALAND, a full-stack real estate marketplace web application integrating AI-powered user guidance with modern web engineering practices. The platform combines a React.js and Tailwind CSS frontend, structured property listing and management workflows, dynamic filtering, and a Claude API-based conversational assistant to deliver a comprehensive property search and transaction support experience. User testing with 200 participants demonstrated a satisfaction rate exceeding 90% and significant improvements in search efficiency and comprehension of property information. The system demonstrates effective integration of large language model APIs within domain-specific web applications and establishes a foundation for further intelligent real estate platform development.

ACKNOWLEDGMENT

I would like to sincerely thank Pooja Sharma, Assistant Professor, Department of Computer Science and Engineering, Raffles University, for her valuable guidance, continuous support, and helpful suggestions throughout this project.

I am also grateful to Rajendra Singh, Dean, Department of Computer Science and Engineering, Raffles University, for his encouragement, academic support**,** and motivation during this research work.

REFERENCES

[1] React Documentation. (2024). React – A JavaScript library for building user interfaces. https://react.dev

[2] Tailwind CSS Documentation. (2024). Utility-First CSS Framework. https://tailwindcss.com

[3] Anthropic. (2024). Claude API Documentation. https://docs.anthropic.com

[4] Ramakrishnan, R., & Gehrke, J. (2003). Database Management Systems (3rd ed.). McGraw-

Hill.
[5] Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill.
[6] Nielsen, J. (1994). Usability Engineering. Morgan Kaufmann.
[7] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
[8] Brown, T., et al. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems, 33, 1877–1901.
[9] Fling, B. (2009). Mobile Design and Development. O'Reilly Media.
[10] Welling, L., & Thomson, L. (2016). PHP and MySQL Web Development (5th ed.). Addison-Wesley.
[11] Flanagan, D. (2020). JavaScript: The Definitive Guide (7th ed.). O'Reilly Media.
[12] Duckett, J. (2011). HTML and CSS: Design and Build Websites. Wiley.
[13] Sommerville, I. (2015). Software Engineering (10th ed.). Pearson.
[14] Welling, L., & Thomson, L. (2016). PHP and MySQL Web Development (5th ed.). Addison-Wesley.